

## cctbx PDB handling tools

Ralf W. Grosse-Kunstleve<sup>a</sup> and Paul D. Adams<sup>a,b</sup>

<sup>a</sup>Lawrence Berkeley National Laboratory, Berkeley, CA 94720

<sup>b</sup>Department of Bioengineering, University of California at Berkeley, Berkeley, CA 94720

Correspondence email: [RWGrosse-Kunstleve@LBL.Gov](mailto:RWGrosse-Kunstleve@LBL.Gov)

### Introduction

The PDB format is the predominant working format for atomic parameters (coordinates, occupancies, displacement parameters, etc.) in macromolecular crystallography. Many small-molecule programs also support this format. The PDB format specifications are available at <http://www.pdb.org/>. Technically, the format is very simple, therefore a vast number of parsers exist in scientific packages. This article is about the PDB handling tools included in the Computational Crystallography Toolbox (cctbx, <http://cctbx.sourceforge.net/>), the open-source component of the PHENIX project (<http://www.phenix-online.org/>).

The evolution of the cctbx PDB handling tools has gone through three main stages spread out over several years. A simple parser implemented in Python has been available for a long time. In many cases Python's runtime performance is sufficient for interactive processing of PDB files, but can be limiting for large files, or for repeatedly traversing the entire PDB database. This has prompted us to implement a fast C++ parser that is described in Grosse-Kunstleve et al. (2006). However, initially the fast cctbx PDB handling tools only supported "read-only" access. Writing of PDB files was supported only at a very basic level. This shortcoming has been removed and the current cctbx version provides comprehensive tools for reading, manipulating, and writing PDB files. These tools are available from both Python and C++, under the iotbx.pdb module.

This article presents an overview of the main types in the iotbx.pdb module, considerations that lead to the design, and related important nomenclature. It is not a tutorial for using the iotbx.pdb facilities. For this, refer to <http://cctbx.sourceforge.net/sbgrid2008/tutorial.html>. See also [http://cci.lbl.gov/hybrid\\_36/](http://cci.lbl.gov/hybrid_36/) which describes iotbx.pdb facilities for handling very large models.

### Real-world PDB files

The simplicity of the PDB format is only superficial and, in the general case, stops after the initial parsing level. The structure of the PDB file implies a *hierarchy* of objects. A first approximation is this hierarchical view is:

```
model
  chain
    residue
      atom
```

This is only an approximation because of a feature that is easily overlooked at first: the "altloc" (official PDB nomenclature) column 17 of PDB ATOM records, specifying "alternative location" identifiers for atoms in alternative conformations. As it turned out, about 90% of the development time invested into iotbx.pdb was in some form related to alternative conformations. Our goal was to provide robust tools that work even for the most unusual (but valid) cases, since this is a vital characteristic of any automated system. The main difficulties encountered while pursuing this goal were:

- Chains with conformers that have different sequences
- Chains with duplicate resseq+icode (residue sequence number + insertion code)
- Conformers interleaved or separated

These difficulties are best illustrated with examples. An old version of PDB entry 2IZQ (from Aug 2008) includes a chain with conformers that have different sequences, the residue with `resseq 11` in chain A, atoms 220 through 283:

```

HEADER      ANTIBIOTIC                               26-JUL-06  2IZQ
ATOM       220  N  ATRP  A  11      20.498  12.832  34.558  0.50  6.03              N
.....ATRP  A  11 .....
ATOM       243  HH2ATRP  A  11      15.522   9.077  38.323  0.50 10.40              H
ATOM       244  N  CPHE  A  11      20.226  13.044  34.556  0.15  6.35              N
.....CPHE  A  11 .....
ATOM       254  CZ  CPHE  A  11      16.789   9.396  34.594  0.15 10.98              C
ATOM       255  N  BTYR  A  11      20.553  12.751  34.549  0.35  5.21              N
.....BTYR  A  11 .....
ATOM       261  CD1BTYR  A  11      18.548  10.134  34.268  0.35  9.45              C
ATOM       262  HB2CPHE  A  11      21.221  10.536  34.146  0.15  7.21              H
ATOM       263  CD2BTYR  A  11      18.463  10.012  36.681  0.35  9.08              C
ATOM       264  HB3CPHE  A  11      21.198  10.093  35.647  0.15  7.21              H
ATOM       265  CE1BTYR  A  11      17.195   9.960  34.223  0.35 10.76              C
ATOM       266  HD1CPHE  A  11      19.394   9.937  32.837  0.15 10.53              H
ATOM       267  CE2BTYR  A  11      17.100   9.826  36.693  0.35 11.29              C
ATOM       268  HD2CPHE  A  11      18.873  10.410  36.828  0.15  9.24              H
ATOM       269  CZ  BTYR  A  11      16.546   9.812  35.432  0.35 11.90              C
ATOM       270  HE1CPHE  A  11      17.206   9.172  32.650  0.15 12.52              H
ATOM       271  OH  BTYR  A  11      15.178   9.650  35.313  0.35 19.29              O
ATOM       272  HE2CPHE  A  11      16.661   9.708  36.588  0.15 11.13              H
ATOM       273  HZ  CPHE  A  11      15.908   9.110  34.509  0.15 13.18              H
ATOM       274  H  BTYR  A  11      20.634  12.539  33.720  0.35  6.25              H
.....BTYR  A  11 .....
ATOM       282  HH  BTYR  A  11      14.978   9.587  34.520  0.35 28.94              H

```

The original atom numbering does not have gaps. Here we have omitted blocks of atoms with constant `resname` and `resseq+icode` to save space.

As of Jun 8 2010, there are 74 files with mixed residue names in the PDB, i.e. only about 0.1% of the files. However, these files are perfectly valid and a PDB processing library is suitable as a component of an automated system only if it handles them sensibly.

An old version of PDB entry 1ZEH (Aug 2008) includes a chain with consecutive duplicate `resseq+icode`, atoms 878 through 894:

```

HEADER      HORMONE                               01-MAY-98  1ZEH
HETATM     878  C1  ACRS      5      12.880  14.021   1.197  0.50 33.23              C
HETATM     879  C1  BCRS      5      12.880  14.007   1.210  0.50 34.27              C
.....ACRS      5 .....
HETATM     892  O1  ACRS      5      11.973  14.116   2.233  0.50 34.24              O
HETATM     893  O1  BCRS      5      11.973  14.107   2.248  0.50 35.28              O
HETATM     894  O   HOH      5      -0.924  19.122  -8.629  1.00 11.73              O
HETATM     895  O   HOH      6      -19.752  11.918   3.524  1.00 13.44              O
HETATM     896  O   HOH      7      -1.169  17.936  -6.103  1.00 12.89              O

```

To a human inspecting the old 1ZEH entry, it is of course immediately obvious that the water with `resseq 5` is not related to the previous residue with `resseq 5`. However, arriving at this conclusion with an automatic procedure is not entirely straightforward. The human brings in the knowledge that water atoms without hydrogen are not covalently connected to other atoms. This is very detailed, specialized knowledge. Introducing such heuristics into an automatic procedure is likely to lead to surprises in some situations and is best avoided, if possible.

In the PDB archive, alternative conformers of a residue always appear consecutively. However, as mentioned in the introduction, the PDB format is also the predominant working format. Some programs produce files with conformers separated in this way (this file was provided to us by a user):

ATOM	1716	N	ALEU	190	28.628	4.549	20.230	0.70	3.78	N
ATOM	1717	CA	ALEU	190	27.606	5.007	19.274	0.70	3.71	C
ATOM	1718	CB	ALEU	190	26.715	3.852	18.800	0.70	4.15	C
ATOM	1719	CG	ALEU	190	25.758	4.277	17.672	0.70	4.34	C
ATOM	1829	N	BLEU	190	28.428	4.746	20.343	0.30	5.13	N
ATOM	1830	CA	BLEU	190	27.378	5.229	19.418	0.30	4.89	C
ATOM	1831	CB	BLEU	190	26.539	4.062	18.892	0.30	4.88	C
ATOM	1832	CG	BLEU	190	25.427	4.359	17.878	0.30	5.95	C
ATOM	1724	N	ATHR	191	27.350	7.274	20.124	0.70	3.35	N
ATOM	1725	CA	ATHR	191	26.814	8.243	21.048	0.70	3.27	C
ATOM	1726	CB	ATHR	191	27.925	9.229	21.468	0.70	3.73	C
ATOM	1727	OG1	ATHR	191	28.519	9.718	20.259	0.70	5.22	O
ATOM	1728	CG2	ATHR	191	28.924	8.567	22.345	0.70	4.21	C
ATOM	1729	C	ATHR	191	25.587	8.983	20.559	0.70	3.53	C
ATOM	1730	O	ATHR	191	24.872	9.566	21.383	0.70	3.93	O
... residues 191 through 203 not shown										
ATOM	1828	O	AGLY	203	8.948	14.861	23.401	0.70	5.84	O
ATOM	1833	CD1	BLEU	190	26.014	4.711	16.521	0.30	6.21	C
ATOM	1835	C	BLEU	190	26.506	6.219	20.135	0.30	4.99	C
ATOM	1836	O	BLEU	190	25.418	5.939	20.669	0.30	5.91	O
ATOM	1721	CD2	ALEU	190	24.674	3.225	17.536	0.70	5.31	C
ATOM	1722	C	ALEU	190	26.781	6.055	20.023	0.70	3.36	C
ATOM	1723	O	ALEU	190	25.693	5.796	20.563	0.70	3.68	O
ATOM	8722	C	DLEU	190	26.781	6.055	20.023	0.70	3.36	C
ATOM	8723	O	DLEU	190	25.693	5.796	20.563	0.70	3.68	O
ATOM	9722	C	CLEU	190	26.781	6.055	20.023	0.70	3.36	C
ATOM	9723	O	CLEU	190	25.693	5.796	20.563	0.70	3.68	O

In this file, conformers A and B of residue 190 appear consecutively, but conformers C and D appear only after conformers A and B of all residues 191 through 203. While this is not the most intuitive way of ordering the residues in a file, it is still considered valid because the original intention is clear. Since it was our goal to develop a comprehensive library suitable for automatically processing files produced by any popular program, we found it important to correctly handle non-consecutive conformers.

### iotbx.pdb.hierarchy

When developing the procedure capable of handling the variety of real-world situations shown above, we strived to keep the underlying rule-set as simple as possible and to avoid highly specific heuristics (e.g. "water is never covalently bound"). Complex rules imply complex implementations, are difficult to explain and understand, tend to lead to surprises, and are therefore likely to be rejected by the community. With this and the real-world situations in mind, we arrived at the following *primary* organization of the PDB hierarchy:

Primary PDB hierarchy:

```

model(s)
  id
  chain(s)
    id
    residue_group(s)
      resseq
      icode
      atom_group(s)
        resname
        altloc
        atom(s)

```

In this presentation the "(s)" indicates a list of objects of the given type. i.e. a hierarchy contains a list of models, each model has an "id" (a simple string) and holds a list of chains, etc.

Comparing with the "first approximation hierarchy" above, the `residue` type is replaced with two new types: `residue_group` and `atom_group`. These types had to be introduced to cover all the real-world cases shown above. The `residue_group` and `atom_group` types are new and unusual. Before we go into the details of these types, it will be helpful to consider the bigger picture by introducing the alternative *secondary* view of the PDB hierarchy:

Secondary view of PDB hierarchy:

```
model(s)
  id
  chain(s)
    id
    conformer(s)
      altloc
      residue(s)
        resname
        resseq
        icode
        atom(s)
```

This organization is probably more intuitive at first. The first two levels (`model`, `chain`) are exactly the same as in the primary hierarchy. Each chain holds a list of `conformer` objects, which are characterized by the `altloc` character from column 17 in the PDB ATOM records. A conformer is understood to be a *complete copy* of a chain, but usually two conformers *share* some or even most atoms. A `residue` is characterized by a unique `resname` and the `resseq+icode`.

The secondary view of the hierarchy evolved in the context of generating geometry restraints for refinement (e.g. bond, angle, and dihedral restraints), where this organization is most useful. It is also the organization introduced in Grosse-Kunstleve et al. (2006), where it was actually the primary organization. While working with the conformer-residue organization, we found that it is difficult to manipulate a hierarchy (e.g. add or delete atoms) in obvious ways. Finally, while developing the automatic generation of constrained occupancy groups for alternative conformations, the conformer-residue organization proved to be unworkable. The main difficulty is that the relative order of residues with alternative conformations is lost in the conformer-residue organization; it is only given indirectly by interleaved residues with shared atoms -- if they exist. As convenient as the conformer-residue organization is for the generation of restraints, it is a hindrance for other purposes.

The names for the new types in the primary hierarchy were chosen not to collide with the secondary view. We could have used "conformer" and "residue" again, just reversed, but there would be the big surprise that one residue has different `resnames`. To send the signal "this is not what you usually think of as a residue", we decided to use `residue_group` as the type name. A residue group holds a list of `atom_group` objects. All atoms in an atom group have the same `resname` and `altloc`. Therefore "resname\_altloc\_group" would have been another plausible name, but we favored `atom_group` since it is more concise and better conveys what is the main content.

#### Detection of residue groups and atom groups

When constructing the primary hierarchy given a PDB file, the processing algorithm has to detect models, chains, residue groups, atom groups and atoms. Most steps are fairly straightforward, but none of the steps is actually completely trivial. For example, what to do if TER or ENDMDL cards are missing? What if residue sequence numbers are not consecutive? The ribosome community widely uses `segid` instead of chain `id` (even though the `segid` column is officially deprecated by the PDB). What if a file

contains both chain id and `segid`? Documenting all our answers in full detail is beyond the scope of this article (and would be more distracting than helpful anyway because the source code is openly available). Therefore we concentrate on the most important rules for the detection of residue groups and atom groups.

Then central conflict we had to resolve was:

- In order to handle non-consecutive conformers, we have to use the `resseq+icode` to find and group related residues.
- However, we cannot use the `resseq+icode` alone as a guide, because we also want to handle chains with duplicate `resseq+icode` (consecutive or non-consecutive).

This lead to a two stage procedure. In the first stage:

- A residue group is given by a block of consecutive ATOM or HETATM records with identical `resseq+icode` columns (SIGATM, ANISOU, and SIGUIJ records may be interleaved), e.g.

ATOM	234	H	ATRP	A	11	20.540	12.567	33.741	0.50	7.24		H
ATOM	235	HA	ATRP	A	11	20.771	12.306	36.485	0.50	6.28		H
ATOM	244	N	CPHE	A	11	20.226	13.044	34.556	0.15	6.35		N
ATOM	245	CA	CPHE	A	11	20.950	12.135	35.430	0.15	5.92		C

However, there is an important pre-condition:

- Unless a sub-block of ATOM records with identical `resname+resseq+icode` columns contains a main-conformer atom (almost "blank `altloc`"), e.g.

HEADER	ANTIBIOTIC RESISTANCE										07-MAY-97	1AJQ	
CRYST1	52.120	65.080	76.300	100.20	111.44	105.81	P	1					1
HETATM	6097	CA	CA	1	5.676	34.115	52.446	1.00	18.50				CA
HETATM	6100	C2	SPA	1	11.860	36.159	33.853	1.00	14.30				C
HETATM	6107	C6	SPA	1	13.085	36.522	34.644	1.00	17.34				C

In this case the sub-block is assigned to a separate residue group.

Within each residue group, all atoms are grouped by `altloc+resname` and assigned to atom groups. The order of the atoms in a residue group does not affect the assignment to atom groups.

After all atom records are assigned to residue groups and atom groups,

- residue groups with identical `resseq+icode`
- that do not contain main-conformer atoms

are merged in the second processing stage. While two residue groups are merged, atom groups with identical `altloc+resname` from the two sources are also merged.

Note that the grouping steps in this process may change the order of the atoms. However, our implementation preserves the relative order of the atoms as much as possible. I.e. in the hierarchy, `resseq+icode` appear in the original "first seen" order, and similarly for `altloc+resname` within a residue group.

### Construction of secondary view of hierarchy

The secondary view of the hierarchy is constructed trivially from the primary hierarchy objects. For each chain, the complete list of `altloc` characters is determined in a first pass. In a second pass, a conformer object is created for each `altloc`, and a second loop over the chain assigns the atoms to each conformer. Main-conformer atoms are assigned to all conformers, atoms in alternative

conformations only to the corresponding conformer. Because of the difficulties alluded to earlier, the conformer and residue objects in the secondary view are "read-only". I.e. all manipulations such as addition or removal of residues, have to be performed on the primary hierarchy. Re-constructing the secondary view after the primary hierarchy has been changed is very fast (fractions of seconds even for the largest files, e.g. 0.22 s for PDB entry 1HTQ with almost one million atoms).

The `iotbx/examples/pdb_hierarchy.py` script shows how to construct the primary hierarchy from a PDB file, and how to obtain the secondary view.

### Nomenclature related to alternative conformations

The `iotbx.pdb` module uses the following nomenclature to describe various aspects of alternative conformations:

- **Main conf. atom** : an atom with
  - a blank `altloc` character
  - and no other atom with the same `name+resname` (but different `altloc`) in the residue group. This second condition is needed for cases like this:

HEADER	HYDROLASE										12-MAR-04	1SO7	
ATOM	2460	CG1	VAL	A	325	-23.284	97.713	15.815	0.66	21.74		C	
ATOM	2461	CG1AVAL	A	325		-23.010	97.616	18.295	0.66	22.88		C	
ATOM	2462	CG2BVAL	A	325		-24.819	96.373	17.146	0.66	22.57		C	

- **Alt. conf. atom** : not main conf.
- **Conformer** : a complete chain with main conf. atoms and alt. conf. atoms with a specific `altloc`.

Note for completeness: it is also possible to obtain conformers of an individual residue group. However, conceptually this is best viewed as a shortcut for first obtaining the conformers of a chain, and then finding the residue of interest in each conformer, ignoring all other residues.

- **Residue** : complete residue in a conformer with main conf. atoms and alt. conf. atoms.

Residues are classified as follows:

- **pure main conf.** : all main conf. atoms
- **pure alt. conf.** : all alt. conf. atoms
- **proper alt. conf.** : both main conf. atoms and alt. conf atoms, all alt conf. atoms have a non-blank `altloc` column
- **improper alt. conf.** : both main conf. atoms and alt. conf atoms, one or more alt conf. atoms have a blank `altloc` column (as shown in the 1SO7 fragment above).

### Errors and warnings

The tools in `iotbx.pdb` are designed to be as tolerant as reasonably possible when processing input PDB files. E.g., as of Jun 8 2010, all 65802 files in the PDB archive can be processed without generating exceptions. However, to assist users and developers in creating PDB files without ambiguities, the hierarchy object provides methods for flagging likely problems as errors and warnings. It is up to the application how to react to the diagnostics.

The `phenix.pdb.hierarchy` command can be used to quickly obtain a summary of the hierarchy in a PDB file and some diagnostics. This example command highlights all main features:

```
phenix.pdb.hierarchy pdb1jxw.ent.gz
```

## Output:

```

file pdb1jxw.ent.gz
total number of:
  models:      1
  chains:      2
  alt. conf.:  5
  residues:    49
  atoms:      786
  anisou:      0
number of atom element+charge types: 5
histogram of atom element+charge frequency:
  " H " 383
  " C " 261
  " O "  77
  " N "  59
  " S "   6
residue name classes:
  "common_amino_acid" 48
  "other"              1
number of chain ids: 2
histogram of chain id frequency:
  " 1 "
  "A" 1
number of alt. conf. ids: 3
histogram of alt. conf. id frequency:
  "A" 2
  "B" 2
  "C" 1
residue alt. conf. situations:
  pure main conf.: 32
  pure alt. conf.: 3
  proper alt. conf.: 14
  improper alt. conf.: 0
chains with mix of proper and improper alt. conf.: 0
number of residue names: 16
histogram of residue name frequency:
  "CYS" 6
  "THR" 6
  "ALA" 5
  "ILE" 5
  "PRO" 5
  "GLY" 4
  "ASN" 3
  "SER" 3
  "ARG" 2
  "LEU" 2
  "TYR" 2
  "VAL" 2
  "ASP" 1
  "EOH" 1   other
  "GLU" 1
  "PHE" 1
### WARNING: consecutive residue_groups with same resid ###
number of consecutive residue groups with same resid: 2
  residue group:
    "ATOM   378  N   PRO A  22 .*.      N  "
    ... 12 atoms not shown
    "ATOM   391  HD3APRO A  22 .*.      H  "
  next residue group:
    "ATOM   392  CB BSER A  22 .*.      C  "
    ... 6 atoms not shown
    "ATOM   399  HB3CSER A  22 .*.      H  "
  -----
  residue group:
    "ATOM   432  N   LEU A  25 .*.      N  "
    ... 18 atoms not shown
    "ATOM   439  CD1CLEU A  25 .*.      C  "
  next residue group:
    "ATOM   452  CG1BILE A  25 .*.      C  "
    ... 13 atoms not shown
    "ATOM   466  HD13CILE A  25 .*.      H  "

```

The diagnostics are mainly intended for PDB working files, but we have tested the `iotbx.pdb` module by processing the entire PDB archive. (This took about 3700 CPU seconds, but using 40 CPUs the last job finished after only 277 seconds. Disk and network I/O is rate-limiting in this case, not the performance of the PDB handling library.) The results are summarized in the following table:

Total number of .ent files: 65802 (2010 Jun 8)

```
56873  WARNING: duplicate chain id
      3  WARNING: consecutive residue_groups with same resid
      2  ERROR:   duplicate atom labels
      1  ERROR:   improper alt. conf.
      0  ERROR:   duplicate model id
      0  ERROR:   residue group with multiple resnames using same altloc
```

About 86% of the PDB entries re-use the same chain id for multiple chains (which are either separated by other chains or TER cards). In many cases, the re-used chain id is the blank character, which is clearly a minor issue. However, we flag this situation to assist people in producing new PDB files with unambiguous chain ids.

The next item in the list, "consecutive residue\_groups with same resid" (where `resid=resseq+icode`), was mentioned before. This situation is best avoided to minimize the chances of mis-interpreting the PDB files.

"duplicate atom labels" pose a serious practical problem (therefore flagged as "ERROR") since it is impossible to uniquely select atoms with duplicate labels, for example via an atom selection syntax as used in many programs (*CNS*, *PyMOL*, *PHENIX*, *VMD*, etc.) or via PDB records in the connectivity annotation section (LINK, SSBOND, CISPEP). Atom serial numbers are not suitable for this purpose since many programs do not preserve them.

Only one PDB entry (1JRT) includes "improper alt. conf." as introduced in the previous section.

There are no PDB entries with two other potential problems diagnosed by the `iotbx.pdb` module. MODEL ids are unique throughout the PDB archive (as an aside: and all MODEL records have a matching ENDMDL record). Finally, in all 74 files with mixed residue names (i.e. conformers with different sequences), there is exactly one residue name for a given altloc.

### Acknowledgments

We gratefully acknowledge the financial support of NIH/NIGMS under grant number P01GM063210. Our work was supported in part by the US Department of Energy under Contract No. DE-AC02-05CH11231.

### References

Grosse-Kunstleve, R.W., Zwart, P.H., Afonine, P.V., Ioerger, T.R., Adams, P.D. (2006). Newsletter of the IUCr Commission on Crystallographic Computing, 7, 92-105.